Creative Programming for Young Minds

. . . on the TI-99/4A ™

```
210 CALL HCHAR(KW,11,128)
220 CALL HCHAR(KW,22,128)
230 CALL HCHAR(13,KW-1,128,K
T+2)
235 CALL CHAR(64,"3C4299A1A1
99423C")
240 R=KW
250 C=12
260 M$=Z2$
270 GOSUB 3990
280 CALL SOUND(150,1500,K1)
290 GOSUB 4280
300 FOR X=K1 TO 7
310 READ A,B,C
320 CALL COLOR(A,B,C)
330 DATA 9,3,3,10,7,16,11,5,
16,16,5,16,12,7,16,15,7,16,1
4,8,8
340 NEXT X
350 DIM P$(12,32)
360 FOR N=K1 TO 25
370 READ C,C$
380 CALL CHAR(C,C$)
>RU
```

Volume IV

CREATIVE
PROGRAMMING INCORPORATED
A SUBSIDIARY OF R.V. WEATHERFORD CO.

CREATIVE Programming for Young Minds

CREATIVE Programming for Young Minds didn't just happen. It represents the harvested fruit of an idea planted several years ago by Dr. Henry A. Taitt. He saw the pressing need for an enrichment program for young children that would help prepare them for the future they would be instrumental in shaping.

It was cultivated by Marilyn Buxton, whose deep interest in early childhood learning enabled her to find ways to teach primary children to program microcomputers.

It was fertilized by Devin Brown, with his lively wit and creative writing style. He gave it the nutrients it needed to appear in printed form to be shared. His shadow is cast over most of the later authors who patterned their style and examples after his original writings.

It was cared for by Howard Smith, Charles Miller, George Kolopanis, Alverta Darding, Lea Ann Hummel, Robin Koch and others, who worked with it in the lab helping to remove the bugs that would stunt its growth.

It was harvested by Nancy Taitt, Marilyn Hoots, Wayne Owens, Diane ZuHone, and others who typed and phoned and talked with people to spread the word and create a market for the final fruit.

And most important of all were the CHILDREN who tried and tested the materials that were produced. They shared their likes and dislikes, and made certain that everything that was included could be done by young minds.

These books were not created by a publisher to be sold to schools, where they would be used on children. They were instead, created from the successes of children, edited by the concerns of parents, and then offered to anyone that wishes to enrich the minds of young children.

If you elect to use these materials, then you assume the responsibility to encourage independent thought, reward creativity, enhance reasoning and logic, and above all, be forever open to alternate ways to solve problems.

If you do this, your own rewards will be found in the faces of the children you serve.

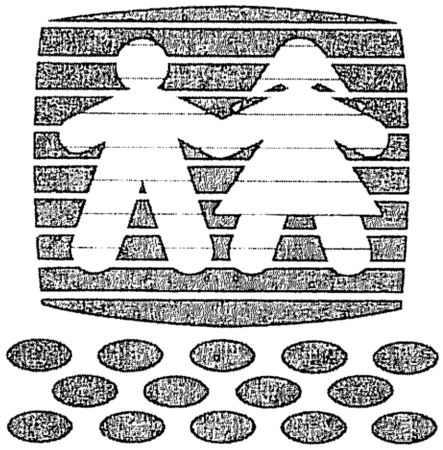
CREATIVE
PROGRAMMING INCORPORATED
A SUBSIDIARY OF R.V. WEATHERFORD CO.

(217) 348-1451

Creative Programming for Young Minds

. . . on the TI-99/4A ™

by Leonard Storm

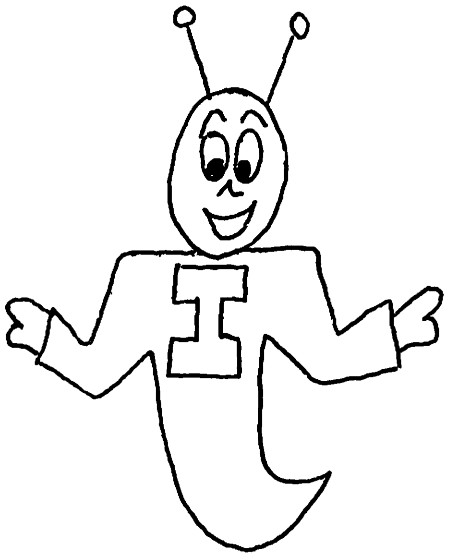


CREATIVE PROGRAMMING FOR YOUNG MINDS

TI-99/4A VOLUME IV

T A B L E   O F   C O N T E N T S

LESSON #13 EDIT



HOWDY, PARTNERS! WELCOME TO VOLUME IV, LESSON 13. IN THIS LESSON, WE WILL BE STUDYING NEW WAYS OF REPAIRING PROGRAM STATEMENTS. (WE ALL MAKE TYPING MISTAKES NOW AND THEN, RIGHT?!) WHEN YOU MAKE CORRECTIONS TO A PROGRAM, YOU ARE EDITING THE PROGRAM.

The commands that you will learn in this lesson will help to take the drudgery out of repairing a program. To begin, type in the following program lines just as they are.

```
10 CALL CLEAR
20 PRINT "LEANING ABOUT COMPURS IS FUNNN!"
30 PRINT "THIS LINE WILL HAVE TO GO!"
40 PRINT "MAKE THIS SOMETHING ELSE."
50 GOTO 20
```

RUN the program, then LIST it.

Now do the following. Type:

```
EDIT 40
```

and then press ENTER .

You have now entered the computer's EDIT mode. This mode will allow you to make changes easily and almost painlessly.

Notice that line 40 has appeared on the TV screen. Now press $\boxed{\text{ENTER}}$ again. The screen scrolls upward one line. Pressing $\boxed{\text{ENTER}}$ has taken you out of the EDIT mode. Whenever you are done making changes to a program, press $\boxed{\text{ENTER}}$ to exit from the EDIT mode.

Now let's go back to the EDIT mode. Type:

```
EDIT 30
```

then press $\boxed{\text{ENTER}}$. Statement 30 should appear on the screen. Now suppose that you didn't want this particular statement in the program. To get rid of it, all you have to do is hold down the $\boxed{\text{FCTN}}$ key and press 3. Do it. Press $\boxed{\text{FCTN}}$ 3. Immediately, the command disappears, but the statement number remains.

Now type in the following command:

```
PRINT "THIS IS SOMETHING ELSE!"
```

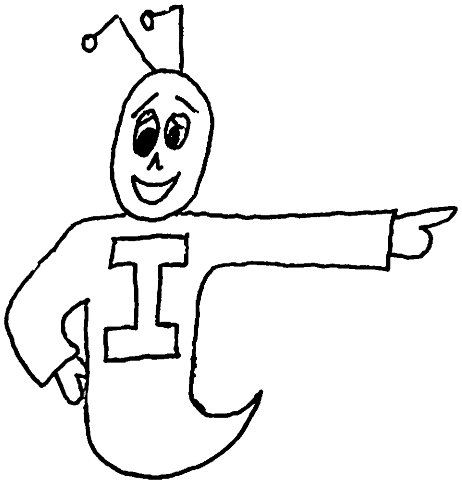
Press $\boxed{\text{ENTER}}$ to exit from the EDIT mode and then LIST the program again. Notice that line 30 has indeed been changed. Now, RUN the program.

Next, stop the program and then type:

```
EDIT 20
```

and press $\boxed{\text{ENTER}}$.

The word FUNNN has a few too many N's. Let's remove the extras. Press the $\boxed{\text{FCTN}}$ key down and hold it down. Now press the ➡ key several times until the cursor has been moved to the first N in FUNNN. (Using $\boxed{\text{FCTN}}$ ➡ allows the cursor to be positioned to a new spot in the line without changing the line in any way.) Finally, press $\boxed{\text{FCTN}}$ 1 twice. This removes two N's. Press $\boxed{\text{ENTER}}$.



IN THE EDIT MODE:

$\boxed{\text{FCTN}}$	3	DELETES A PROGRAM LINE.
$\boxed{\text{FCTN}}$	1	DELETES A SINGLE CHARACTER.

Again, LIST the program to see what changes have resulted.

This time, we will enter the EDIT mode in a much simpler way. Type the number 20 and then press $\boxed{\text{FCTN}}$ ⬆. This also causes line 20 to be printed on the screen.

Now use $\boxed{\text{FCTN}}$ ➡ to position the cursor over the R in COMPURS. Next, press $\boxed{\text{FCTN}}$ 2. ($\boxed{\text{FCTN}}$ 2 allows us to insert characters between other characters.) Now type in the letters TE.

Next, use $\boxed{\text{FCTN}}$ ← to move the cursor to the N in LEANING. Again, press $\boxed{\text{FCTN}}$ 2. They type R. The R is automatically inserted in the proper place. Statement 20 should now look like this:

```
     20 PRINT "LEARNING ABOUT COMPUTERS IS FUN!"
```

Next, press $\boxed{\text{FCTN}}$ ↓ . This causes the changes that you have made to become permanent and also causes the next higher line number to appear for editing. Let's not change statement 30, but go on to statement 40. Press $\boxed{\text{FCTN}}$ ↓ once more.

Now, use $\boxed{\text{FCTN}}$ → to move the cursor to the first letter of MAKE. Then use $\boxed{\text{FCTN}}$ 1 to delete MAKE. Next, use $\boxed{\text{FCTN}}$ 2 to help you insert COMPUTERS in front of the word THIS. Finally, delete the word THIS and replace it with ARE. Statement 40 should now look like this:

```
     40 PRINT "COMPUTERS ARE SOMETHING ELSE."
```

Use $\boxed{\text{FCTN}}$ ↓ two more times. Notice what happens when you run out of program statements. (This causes you to exit from the EDIT mode.)

Now LIST the program to observe the changes that have been made in the program. RUN the program.

Remember that when you press $\boxed{\text{FCTN}}$ ↓ to go to the next higher statement number, any changes you have made in the present statement become permanent. The same will be true if you use $\boxed{\text{FCTN}}$ ↑ to go to lower numbered statements.

Stop the program. Next, type the number 10 and press $\boxed{\text{FCTN}}$ ↑ to enter the EDIT mode. Then use the $\boxed{\text{FCTN}}$ → and FCTN 1 to delete the space between CALL and CLEAR. The statement should finally look like this:

```
10 CALLCLEAR
```

Now press $\boxed{\text{FCTN}}$ 4 and see what happens. The computer leaves the EDIT mode.

Now LIST the program. Notice that the change that was made on the CALL CLEAR line has disappeared. Thus, any changes that have been made on a line will disappear if $\boxed{\text{FCTN}}$ 4 is used to exit from that line. Changes already made permanent in other lines are not affected.

The EDIT functions are summarized below for your convenience:

FUNCTION	COMMENT
EDIT (line number) (line number) $\boxed{\text{FCTN}}$ ↑ (line number) $\boxed{\text{FCTN}}$ ↓	Three ways of entering the EDIT mode.
$\boxed{\text{FCTN}}$ ↑	All changes made to the program line are made permanent. The next lower line number is displayed. If no lower line number exists, the computer leaves the EDIT mode.

FUNCTION	COMMENT
FCTN ↓	All changes made to the program are made permanent. The next higher line number is displayed. If no higher line number exists, the computer leaves the EDIT mode.
ENTER	All changes made to the current program line are made permanent. The computer leaves the EDIT mode. Note, the cursor does not have to be at the end of the line for all of the line to be entered.
FCTN →	Moves the cursor one space to the right.
FCTN ←	Moves the cursor one position to the left.
FCTN 2	Inserts one or more characters.
FCTN 1	Deletes one character.

FUNCTION	COMMENT
$\boxed{\text{FCTN}}$ 4	Causes the screen to scroll up one line. The computer leaves the EDIT mode. Any changes made on the line before $\boxed{\text{FCTN}}$ 4 was pressed are ignored.
$\boxed{\text{FCTN}}$ 3	Erases the current program line but not the statement number.

Finally, try typing the number 15 and pressing $\boxed{\text{FCTN}}$ ⬆.
What error statement results? ______________________________

Another convenient feature of your TI home computer is the NUMBER command. This command allows you to number program lines without actually typing in every line number.

Do the following: Enter NEW, then type NUMBER and press $\boxed{\text{ENTER}}$. 100 should appear on the screen. Now enter the following program lines:

(Press $\boxed{\text{ENTER}}$ after each line.)

```
A$="HELLO"
B$="I'M A TEXAS INSTRUMENTS"
C$="HOME COMPUTER."
PRINT A$
```

Keep going. ⟶

```
PRINT B$

PRINT C$

GOTO 160
```

(Press ENTER *one more time.)*

Notice how one exits from the line numbering command by hitting the ENTER key one extra time.

RUN the program.

The first statement of the program contains the string variable, A$. String variables may be given values by using the equal sign. However, string variables may not be assigned numerical values, unless they are enclosed in quotes. Inside quotes, either letters or numbers may be used.

For example, use the EDIT mode to change statement 100 to
A$ = 10.
Will the program RUN?

Change statement 100 to A$ = "10". Now RUN the program again. This time the program runs because A$ has been given a string value of "10". When numbers are enclosed inside quotes, they are treated as string constants and not as number constants.

But to get back to the number command . . .

The general form of the NUMBER command is shown below:

NUMBER *start,step*

Start is a number which specifies the line number of the first program line.

Step is a number which tells the computer how far apart the line numbers will be.

For example:

NUMBER 40,5 would number the program lines: 40, 45, 50, 55, etc.

NUMBER 200,20 would number the program lines: 200, 220, 240, etc.

NUMBER 30 would number the lines: 30, 40, 50, 60, etc. (Step = 10 is understood.)

NUMBER ,5 would number the lines: 100, 105, 110, 115, etc. (Start = 100 is understood.)

What NUMBER command would number program lines by three's starting with 90? ________________________________

EXERCISE 13-1

Type in the following program. Use the NUMBER command. Then use the EDIT functions to make the corrections listed below. When you are through, LIST the program to check that all of the corrections have been made.

```
10 A$="FOOXURRXSCOOOREYAND"
20 B$="SETVNENXYEARSAGO"
30 PRINT A$;" ";B$
40 GOTO 40
```

In line 10, delete the following characters indicated by the arrows:

```
         ↓↓  ↓↓   ↓↓  ↓
10 A$="FOOXURRXSCOOOREYAND"
```

Now 10 looks like this:

```
10 A$=FOURSCOREAND"
```

Next insert a space between the three words FOUR, SCORE, and AND.

Finally, repair statement 20 so that it reads:

```
20 B$="SEVEN YEARS AGO."
```

LESSON #14 GOSUB and RETURN

In this lesson, we will be learning about parts of programs called subroutines. Subroutines are portions of a program which are used several times during program execution.

The following example shows how subroutines may be used in a program. Type the program into the computer and then RUN it.

```
5 CALL SCREEN(7)
10 CALL CLEAR
20 CALL COLOR(2,3,16)
30 CALL CHAR(40,"FFFFFFFFFFFFFFFF")
4Ø CALL CHAR(41,"ØØØØØØØØØØØØØØØØ")
50 FOR I=9 to 23
60 CALL VCHAR(5,I,41,17)
70 NEXT I
80 ROW=6
90 COL=10
100 GOSUB 1000
110 COL=19
120 GOSUB 1000
130 ROW=8
140 COL=5
150 GOSUB 1000
160 COL=24
170 GOSUB 1000
```

Keep going. ⟶

```
180 ROW=12

190 COL=13

200 GOSUB 1000

210 COL=16

220 GOSUB 1000

230 ROW=17

240 COL=10

250 GOSUB 1000

260 COL=13

270 GOSUB 1000

280 COL=16

290 GOSUB 1000

300 COL=19

310 GOSUB 1000

320 GOTO 320

999 REM  PRINT THE CIRCLE

1000 CALL HCHAR(ROW,COL+1,40,2)

1010 CALL VCHAR(ROW+1,COL+3,40,2)

1020 CALL VCHAR(ROW+1,COL,40,2)

1030 CALL HCHAR(ROW+3,COL+1,40,2)

1040 RETURN
```

RUN the program.

This is how the program works:

Statements 1000 through 1040 are the hard-working statements which make up the subroutine. This subroutine causes a "circle" to be printed on the screen at a location which depends on the value of ROW and COL.

Statements 20 through 40 define the characters to be printed. Statements 50 through 70 print a white background which forms the head of the robot figure. Then in 80 and 90, ROW and COL are set to values which represent a position on the screen. In statement 100, the program tells the computer to <u>go</u> to the <u>sub</u>routine (GOSUB). So the next statement which gets executed is statement 1000. When the computer gets to statement 1040, the program tells it to RETURN to the main program. The computer then jumps back to the statement following the GOSUB statement (line 110).

Note that GOSUB statements at lines 100, 120, 150, 170, 200, 220, 250, 270, 290, and 310 all cause a jump to statement 1000. Then when the subroutine is completed, the RETURN causes the computer to return to the main program at the line after the GOSUB statement which called the subroutine. This is unlike a GOTO statement which must jump to the same point in a program each time it is used.

Now add the following lines to your program

```
     1035 GOSUB 2000
     1999 REM  SOUND SUBROUTINE
     2000 CALL SOUND(100,262,0)
     2010 RETURN
```

RUN the program again to see what happens.
Here's how the program works now. When statement 100 is executed, the program jumps to statement 1000. Statements 1000 through 1030 cause a "circle" to be printed on the screen. Then statement 1035 causes a jump to the SOUND subroutine at statement 2000. Statement 2010 causes a RETURN from the SOUND subroutine to the first subroutine at statement 1040. But 1040 causes a RETURN to statement 110.

To further illustrate the GOSUB and RETURN statements, type in the following program:

```
     10 PRINT "A GOSUB DEMONSTRATION"
     15 PRINT "GOTO ROUTINE AT 1000"
     20 GOSUB 1000
     25 PRINT "GOTO ROUTINE AT 2000"
     30 GOSUB 2000
     35 PRINT "GOTO ROUTINE AT 3000"
     40 GOSUB 3000
     45 PRINT "END"
     50 GOTO 50
     1000 PRINT "SUBROUTINE #1 AT 1000"
```

Keep going. ⟶

```
1010 PRINT "FROM SUBROUTINE 1 TO 2"
1020 GOSUB 2000
1030 PRINT "RETURN FROM SUBROUTINE #1"
1040 RETURN
2000 PRINT "SUBROUTINE #2 AT 2000"
2010 PRINT "FROM SUBROUTINE 2 TO 3"
2020 GOSUB 3000
2030 PRINT "RETURN FROM SUBROUTINE #2"
2040 RETURN
3000 PRINT "SUBROUTINE #3 AT 3000"
3010 PRINT "RETURN FROM SUBROUTINE #3"
3020 RETURN
```

Try to figure out in what order the program statements will be executed before you try running the program. Then RUN the program to check your answer.

On the lines below, record the order in which the statements were executed.

10, 15, 20, 1000, ____________________

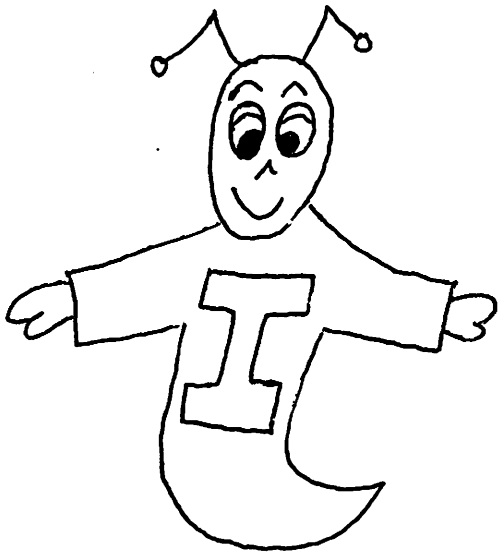
____

____

____

____

Subroutines are a very useful part of long programs because subroutines allow the programmer to break a program down into specific tasks. Each task can be accomplished using a specific subroutine. Thus, if a certain task has to be done many times, instead of writing the commands over and over to do the task, a subroutine can be called each time the task is to be performed.



GOSUB STATEMENTS CAUSE THE COMPUTER TO "REMEMBER" ITS POSITION IN THE CALLING PROGRAM AND THEN JUMP TO THE SPECIFIED SUBROUTINE.

ALL SUBROUTINES MUST END WITH A RETURN STATEMENT WHICH CAUSES THE COMPUTER TO RETURN TO THE "REMEMBERED" POSITION IN THE CALLING PROGRAM.

EXERCISE 14-1

Each of the following programs contain errors connected with the GOSUB command. RUN each of the programs and record the error statement which results.

```
10 PRINT "RETURN STATEMENT"
20 PRINT "OCCURS BEFORE A"
30 PRINT "GOSUB STATEMENT"
40 RETURN
```

____

```
10 GOSUB 10
```

____

```
10 GOSUB 20
```

____

EXERCISE 14-2

Write a program that INPUTS a number, A, from the keyboard and then uses a subroutine to print that number A, A times in a row. The program should then go back to the beginning. Test your program by RUNning it.

EXERCISE 14-3

String variables were introduced earlier in this lesson. In this exercise, you will learn some ways in which string symbols may be manipulated and compared.

Type in the following program and RUN it. Notice the effect of the & operator.

```
10 A$="ABC"
20 B$="DEFGHIJ"
30 PRINT B$
40 PRINT A$
50 PRINT A$ & B$
```

What does statement 50 print? ______________________________

The & operator causes string constants to be concatenated or joined together.

String symbols may be compared using the following relational operators:

= , < , > ,<> , <=, >= .

Note that a string expression may only be compared with a string expression, never a numeric expression. String expressions are compared character by character from left to right using the character codes listed in an earlier manual. Thus, a list of string expressions can be sorted and put into alphabetical order.

Type in the following program and then RUN it. This program illustrates how string expressions may be compared alphabetically.

```
10 PRINT "INPUT ONE WORD"
20 INPUT A$
30 PRINT "INPUT A SECOND WORD"
40 INPUT B$
50 IF A$<B$ THEN 110
60 IF A$ = B$ THEN 90
70 PRINT B$;" COMES BEFORE ";A$
80 GOTO 10
90 PRINT B$;" IS IDENTICAL TO ";A$
100 GOTO 10
110 PRINT A$;" COMES BEFORE ";B$
120 GOTO 10
```

INPUT the following words in the above program. Then record the program's response.

FIRST WORD	SECOND WORD	PROGRAM RESPONSE
A	I	______________________
AT	AS	______________________
AS	ASK	______________________
YOU	YOU	______________________
XYZ	XYY	______________________

Now try a few on your own.

FIRST WORD	SECOND WORD	PROGRAM RESPONSE

In the case of "AS" and "ASK", the computer compares the first character on the left of each string and "sees" that they are equal. The computer then compares the second characters from the left and finds them equal. The computer searches this way until it finds different characters or until it runs out of characters in the expression. Since "ASK" contains an extra letter, the computer "decides" that "ASK" > "AS". So, "AS" comes before "ASK".

LESSON #15 SOUND

The TI-99/4A home computer has a music capability already built in. This capability is accessed by using the CALL SOUND command. This command is capable of producing single tones as well as chords consisting of three separate tones. In this lesson, you will learn how to use the CALL SOUND command to make your own music.

Type in the following command, then press $\boxed{\text{ENTER}}$.

```
     CALL SOUND(1000,440,2)
```

Note the tone that is produced. (If you don't hear a tone, turn up the volume setting on your TV screen.)

Now observe the change which occurs when you change the 1000 to 4250. That is, type in:

```
     CALL SOUND(4250,440,2)
```

The same tone is again produced, but now it lasts much longer. The first number in the parenthesis controls duration of the tone. That is, it controls how long the tone lasts. The number representing the duration must be in the range from 1 to 4250, inclusive (or from -1 to -4250, inclusive). Each 1000 represents 1 second of the time so that the duration ranges from .001 to 4.25 seconds.

In a program, the computer will continue to execute program statements while a sound is being played. However, the computer will not execute a second CALL SOUND statement until the first sound has been completed unless a negative duration is specified. If you specify a negative duration in the second CALL SOUND statement, then the first sound will be stopped and the second sound will be started immediately when the computer encounters the second CALL SOUND statement.

The following two programs illustrate this property. Type the first in and RUN it. Then type the second program in and RUN it. Notice the timing difference.

```
10 CALL CLEAR
20 CALL SOUND(4250,440,2)
30 PRINT "FIRST SOUND"
40 CALL SOUND(2000,880,2)
50 PRINT "SECOND SOUND"
60 GOTO 20
```

In the above program, statement 20 starts the first sound. This sound will last about 4.25 seconds. Statement 30 is executed while the first sound is playing. However, statement 40 is not executed until after the first sound stops.

To form the second program, change statement 40 to:

```
40 CALL SOUND(-2000,880,2)
```

Now RUN the program.

Again, statement 20 starts the first sound. Statement 30 is executed during this sound. But now, statement 40 is executed immediately after statement 30 without waiting for the 4.25 second interval of statement 20 to end.

The second number in the CALL SOUND command controls the frequency of the tone, that is, its pitch. The number representing the tone may range from 110 to 44733, inclusive. This number is the frequency of the tone in cycles per second or Hertz. The frequencies for some common musical notes are given on pages 176 and 177.

The program below will allow you to easily produce tones on the computer. Type it into the computer, then RUN it.

```
10 CALL CLEAR
20 INPUT "ENTER A NUMBER FROM 110 TO 44733 ":X
30 CALL SOUND(2000,X,0)
40 GOTO 10
```

Type in as many numbers as you wish and listen to the change in tone.

Also, try the following sequence of frequencies:

220, 247, 262, 294, 330, 349, 392, 440

The third number in the CALL SOUND command tells the computer how loud to make the sound. This number may range from 0 (the loudest) to 30 (the quietest), inclusive.

Type the following program into the computer and RUN it. It illustrates the use of the volume control number.

```
10 CALL CLEAR
20 FOR VOL=30 TO 0 STEP -1
30 CALL SOUND(1000,262,VOL)
40 PRINT "VOL=";VOL
50 NEXT VOL
60 GOTO 20
```

In addition to producing musical tones, the CALL SOUND command can be used to generate 8 different noises. This is done by using the numbers from -1 to -8, inclusive, for the frequency argument of the CALL SOUND command.

Type the following program into the computer and RUN it.

```
10 CALL CLEAR
20 FOR NOISE= - 8 TO -1
```

Keep going. ⟶

```
30 CALL SOUND(1000,NOISE,2)
40 PRINT"NOISE=";NOISE
50 NEXT NOISE
60 GOTO 10
```

The characteristics of each noise type are shown in the table below:

FREQUENCY NUMBER	NOISE TYPE
-1	Periodic 1
-2	Periodic 2
-3	Periodic 3
-4	Variable periodic
-5	White noise 1
-6	White noise 2
-7	White noise 3
-8	Variable white noise

The noise types variable periodic and variable white noise have some special properties which will be explained a little later.

Up to three different tones and one noise may be played simultaneously using one CALL SOUND command. The order of the arguments is shown below:

```
CALL SOUND(DURATION,TONE 1,VOLUME 1,TONE 2, VOLUME 2,
           TONE 3,VOLUME 3,NOISE,VOLUME)
```

Note that all the tones must have the same duration, but each tone may be given a different volume.

The program below shows how tones may be blended together. Type in the program and RUN it.

```
10 CALL CLEAR
20 CALL SOUND(4000,262,0)
30 PRINT "THIS IS MIDDLE C"
40 CALL SOUND(4000,262,0,330,0)
50 PRINT "THIS IS C+E"
60 CALL SOUND(4000,262,0,330,0,392,0
70 PRINT "THIS IS C+E+G"
80 CALL SOUND(4000,262,0,330,5,392,10,-5,10)
90 PRINT "C+E+G+NOISE"
```

Now let's get back to the variable noise types; those with frequency values -4 and -8. These two noise types can be varied by altering the frequency of the third tone in a CALL SOUND command. This property is illustrated in the program listed on the next page. Type it in and RUN it. Notice the change in the noise as the frequency of the third tone changes.

```
10 CALL CLEAR
20 LET V3=15
30 LET V=30
40 T=110
50 NOISE=-4
60 GOSUB 100
70 LET NOISE=-8
80 GOSUB 100
90 GOTO 90
100 FOR I=1 TO 100
110 T3=I*110
120 CALL SOUND(250,T,V,T,V,T3,V3,NOISE,0)
130 NEXT I
140 RETURN
```

Notice statements 20, 30, and 70. These statements show an alternate way to assign values to variables.

Statements 20 through 50 assign numerical values to the variables which will be used in the CALL SOUND statement. Once these values are set, the program jumps to a subroutine at statement 100. This subroutine causes the CALL SOUND statement to be executed 100 times with increasing third tone, T3. Varying the third tone causes the NOISE sound to change. The first two tones are given a volume level of 30 so that they will not be heard.

After these 100 noise tones, the computer jumps back to line 70 which sets the next type of variable noise which will be used in the CALL SOUND statement. Statement 80 then causes a jump to the subroutine where the new noise type is sounded 100 times.

A partial list of frequencies and musical notes is given below. The table can be extended quite simply, however, by using the program listed on the page following the table.

FREQUENCY	NOTE	FREQUENCY	NOTE
110	A	262	C(middle C)
117	A#	277	C#
123	B	294	D
131	C	311	D#
139	C#	330	E
147	D	349	F
156	D#	370	F#
165	E	392	G
175	F	415	G#
185	F#	440	A
196	G	446	A#
208	G#	494	B
220	A	523	C
233	A#	554	C#
247	B	587	D

Keep going. ⟶

FREQUENCY	NOTE	FREQUENCY	NOTE
622	D#	1109	C#
659	E	1175	D
698	F	1245	D#
740	F#	1319	E
784	G	1397	F
831	G#	1480	F#
880	A	1568	G
932	A#	1661	G#
988	B	1760	A
1047	C		

NOTE: A# = B^b C# = D^b F# = G^b

```
10 REM  THIS IS A PROGRAM
20 REM  WHICH CALCULATES A
30 REM  FREQUENCY GIVEN THE
40 REM  NUMBER OF THE NOTE
50 INPUT "INPUT NOTE NUMBER": N
60 FREQ=110*(2∧(1/12))∧N
70 PRINT "N=";N,"FREQ=";FREQ
80 GOTO 50
```

Type the above program into the computer's memory and then RUN it.

Input the following note numbers and record the results.

N	FREQ
0	A is 0 notes away from 110
1	A# is 1 note away from 110
2	B is 2 notes away from 110
3	etc.
12	
15	
24	
36	
48	
49	

Notice that the notes always follow the same order: A, A#, B, C, C#, D, D#, E, F, F#, G, G#, then the note names repeat.

Therefore, if N = 48 corresponds to the highest A note in the table, what note does N = 49 yield? _______________
N = 50? _______________

Also, observe something else about the musical scale. All the A notes are related by factors of 2. For example:

low A = 110

next A = 2 X 110 = 220

A above middle C = 220 X 2 = 440

etc.

The same relationship holds for any of the notes. For example:

low C = 131

middle C = 2 X 131 = 262

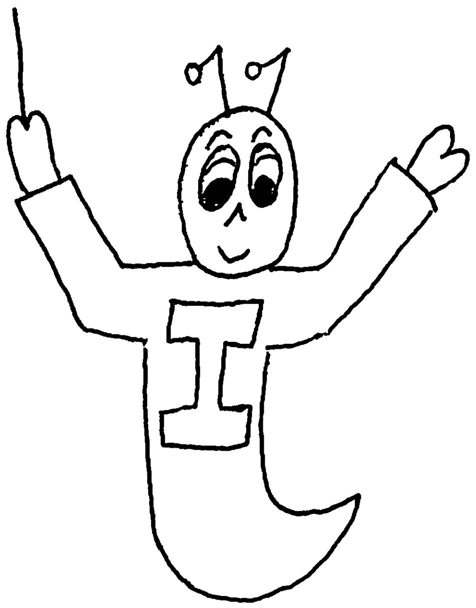
high C = 2 X 262 = 524

etc.

Using this relationship, what is the frequency of the first C-note above the 1047 Hertz C? ____________________

What is the frequency of the first D# note above the 1245 Hertz D#? ________________________

EXERCISE 15-1



NOW IT'S YOUR TURN TO WRITE THE MUSIC! JUST REMEMBER THE CORRECT FORM FOR THE CALL SOUND COMMAND. ONE DURATION NUMBER IS REQUIRED. FROM ONE TO THREE FREQUENCIES MAY BE SPECIFIED AND ONE NOISE. EACH SPECIFIED FREQUENCY MUST BE FOLLOWED BY A VOLUME. THE CORRECT FORM IS REPEATED BELOW:

```
CALL SOUND(DURATION,TONE 1,VOLUME 1,
TONE 2,VOLUME 2,TONE 3,VOLUME 3,
NOISE,VOLUME)
```

Write a program that sets the variable, TONE, equal to each of the numbers in the list on the next page (one at a time, of course). After TONE is set equal to one of the numbers, the program should jump to a subroutine which plays the note which TONE represents. Then TONE is set equal to the next number. The new tone is played, and so on. Test your program by RUNning it.

LIST OF TONES: 220, 247, 262, 294, 330, 349, 392, 440

EXERCISE 15-2

Write a program which causes a small missile to move down the screen and strike a small target. When the missile hits the target, an exploding-like noise should occur and the target should change to a flash:

EXERCISE 15-3

Write a program that plays all the notes from 110 Hertz to 1760 Hertz by using the formula given in statement 60 on page 177.

EXERCISE 15-4

Fill in the following blanks with the correct responses to the following problems.

1. Write a CALL SOUND command that will play the frequencies 220 Hz, 262 Hz, and 349 Hz all at the same time, with the same loudness, and for a duration of 1 second.

____

Type the command into the computer and listen to the sound.

2. Write a CALL SOUND command that would play the following notes simultaneously for two seconds, all with the same volume: middle C, F above middle C, and A above middle C.

____

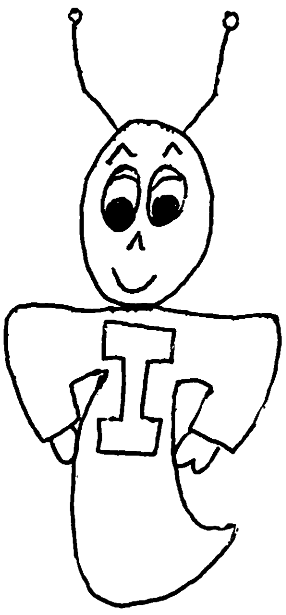
Again, let the computer play the notes for you.

3. Write a command that will play the following notes for 1.5 seconds. Give each note the indicated relative volume.

349 Hz	maximum volume
440 Hz	softer
523 Hz	softest, but audible

____

LESSON #16 LIST "TP"



IN THIS LESSON, YOU WILL LEARN HOW TO LIST YOUR PROGRAMS USING THE THERMAL PRINTER PLUGGED INTO THE RIGHT SIDE OF YOUR TI 99/4 HOME COMPUTER. BY USING THE THERMAL PRINTER, YOU CAN OBTAIN WRITTEN COPIES OF ANY OF YOUR PROGRAMS. THIS CAN BE VERY HELPFUL WHEN YOU NEED TO REVISE A PROGRAM.

To begin, check to see that the Thermal Printer still has a supply of thermal paper. Lift the black plastic lid located on top of the printer to access the paper compartment. If the paper roll is nearly empty, tell your instructor.

If the supply of paper is adequate, close the paper compartment and turn on the computer (if it is not already on). Then turn on the Thermal Printer by sliding the switch to the right. It is located on the front of the printer at the bottom right corner.

Next, try pressing the paper advance button briefly to see that the printer is on. If it is, the paper will begin to advance. You are now ready to do some listing.

First, type the following program into the computer.

```
10 REM  THERMAL PRINTER TEST
20 REM  ABCDEFGHIJKLMNOPQRSTUVWXYZ
30 REM  1234567890
40 REM  !@#$%¢'&*()
50 REM  >_-+"<∧/=?:;,
```

Next, type the following into the computer and press $\boxed{\text{ENTER}}$.

```
LIST "TP"
```

The Thermal Printer should eject 5 lines of paper, print the program lines on the thermal paper, then eject 5 more lines of paper.

Now, if you want, the program listing may be torn off and saved. Notice that the Thermal Printer has a cutting edge for this purpose.

The Thermal Printer has many other features. You will learn about some of them in this lesson. The more advanced topics will show up in later lessons.

Besides the capability of listing an entire program, one may also list just a part of a program. For example, type in the following command:

```
LIST "TP": 20-40
```

The above command causes only the program lines 20, 30, and 40 to be listed.

Now type in the command:

```
LIST "TP":40
```

This command lists only the statement numbered 40.

Perhaps you are thinking that this 5 line ejection business is wasteful of paper. Well, you do have the option to suppress the ejection by using something called a _software switch_. A switch is usually thought to be a piece of hardware (something physical) which can be used to turn something on or off. A light switch is an example. A _software switch_ is a non-physical switch, a programmed switch which can turn something on or off, for example, paper ejection.

Type in the following command and press $\boxed{\text{ENTER}}$.

```
LIST "TP.E"
```

The .E is the software switch which suppresses extra paper ejection.

Now try this:

```
LIST "TP.S"
```

What does the .S software switch do? ______________________

____

Finally, try this command:

```
LIST "TP.S.E"
```

For practice, pick out one of the programs that you have written and stored on tape. Use the OLD command to put the program into the computer's memory. Then use the LIST "TP" command to get a written copy of the program.

____
name

VOLUME IV REVIEW QUIZ

Fill in the blank with the correct word from the ANSWER POOL.

(If you get stuck, turn back to the correct page and review.)

ANSWER POOL

$\boxed{\text{FCTN}}$ 4	CALL SOUND(2620,262,0 330,0,392,0,-5,0)	$\boxed{\text{ENTER}}$
string variables	&	4250 or -4250
$\boxed{\text{FCTN}}$ ⬆	NUMBER 200,20	GOSUB 3000
RETURN	$\boxed{\text{FCTN}}$ 1	90 $\boxed{\text{FCTN}}$
$\boxed{\text{FCTN}}$ 3	EDIT 90	LIST "TP":60-80
negative duration	44733	$\boxed{\text{FCTN}}$ 2

1. ____________________ and ____________________ are two ways to enter the EDIT mode at statement 90 (p.148,150).

2. ____________________ erases a program line, but keeps the statement number, once you are in the EDIT mode (p.149).

3. ____________________ causes a character to be deleted (p.150).

4. ____________________ allows one or more characters to be inserted between other characters (p.150).

5. ____________________ can be used to move from one statement number to the next lower statement number once you are in the EDIT mode (p.151).

6. ____________________ and ____________________ are two ways to leave the EDIT mode (p.152,153).

7. ____________________ would cause automatic numbering starting at 200 with a step of 20 (p.154).

8. ____________________ may not be assigned numerical values, unless enclosed in quotes (p.155).

9. ____________________ would cause the computer to jump to a subroutine at statement 3000 (p.158).

10. ____________________ is the statement every subroutine must end with (p.163).

11. ____________________ causes string constants to be concatenated or joined together (p.166).

12. ____________________ would cause the following tones to be sounded for 2.6 seconds, all at the same maximum loudness: 262, 330, 392, and type 5 noise (p.173).

13. ____________________ in a CALL SOUND statement causes the CALL SOUND statement to be executed as soon as it is encountered in a program. The previous sound is stopped (p.170).

14. ____________________ is the highest frequency which the computer can produce (p.171).

15. ____________________ is the longest sound duration which the computer will produce (p.169).

16. ____________________ would cause statements 60 through 80 of the program currently stored in computer memory to be printed on the Thermal Printer (p.186).

THE COLORED PAGES

At the end of this manual, you will find several colored pages. These are projects that test your ability to use what you have learned. There are no right or wrong answers. If your program does what is asked, then it is quite acceptable. You are free to express your creativity. Be proud of what you do. Do not worry whether your solution is like anyone else's.

Some of these projects may seem easy...but do not be deceived into thinking that you can skip them. After all, if they are easy for you, then it will not take long to do them.

Good luck!

Henry A. Taitt

Henry A. Taitt
Director

GREEN PROJECT 1

EDIT this program so that it will RUN on your computer. Then save it on tape. (You may need to make changes!)

```
10 CALL CLEAR

20 G = 7: N = 7

30 PRINT "YOU ARE THE CAPTAIN OF A SUB-HUNTING DESTROYER."

40 PRINT "YOU HAVE 7 DEPTH CHARGES.  YOUR SONAR WILL"

50 PRINT "TELL YOU WHERE YOUR CHARGES EXPLODE WITH RESPECT"

60 PRINT "TO THE SUB.  GOOD LUCK!"

70 A = INT(7*RND(0):B = INT(7*RND(0);C = INT(7*RND(0))

80 FOR D = 1 TO N:PRINT "TRIAL #"D:INPUT X,Y,Z

90 IF ABS(X-A) + ABS(Y-B) + ABS(Z-C) = 0 THEN 3ØØ

100 GOSUB 400:PRINT:NEXT 0

110 PRINT:PRINT "YOU BLEW IT: YOU'VE BEEN TORPEDOED!"

120 PRINT "THE SUB WAS AT "A", "B", "C"

300 PRINT "    ! ! BOOM ! !  YOU FOUND IT IN "O" TRIES."

400 PRINT "SONAR REPORTS SHOT WAS"

410 IF Y > B THEN PRINT "NORTH-"

420 IF Y < B THEN PRINT "SOUTH-"

430 IF X > A THAN "EAST"

440 IF X < A THEN "WEST"

450 IF Y<>B OR X<>A THEN PRINT "AND"

460 IF Z > C THEN PRINT TOO LOW

470 IF Z < C THEN PRINT TOO HIGH

480 IF Z = C THEN PRINT "DEPTH O.K.!"

490 RETURN

500 NEXT

510 END
```

GREEN PROJECT 2

This program may have errors. It was NOT designed for your computer. See if you can make it RUN on your computer.

```
5 HOME
10 REM ***THIS IS A NUMBER GUESSING GAME***
20 X=100*RND(1)
30 PRINT "I HAVE A NUMBER IN MY MEMORY BETWEEN 100 AND 1"
40 PRINT
45 PRINT "YOU MAY TRY AND GUESS IT!"
46 FOR T=1 TO 500:NEXT T
50 N=N+1
55 HOME
60 PRINT "WHAT IS YOUR GUESS?"
70 INPUT G
80 IF X=G THEN 200
90 IF X=G THEN 290
95 FOR T=1 TO 300:NEXT T
100 PRINT "MY NUMBER IS GREATER THAN YOUR GUESS!"
110 GOTO 50
200 PRINT "YOU SLY FOX, YOU GUESSED IT!"
210 PRINT "IT TOOK YOU";N;"GUESSES."
220 END
```

GREEN PROJECT 3

Take PROJECT 1 and improve it by adding graphics, and making the instructions and logic better. Add sound effects.

GREEN PROJECT 4

CREATE a program that uses 4 subroutines. Make two of them with color graphics and two of them with sound. Let the main program allow you to select which subroutine you wish to use.

GREEN PROJECT 5

If you have a thermal printer, CREATE a sign of your own design.

GREEN PROJECT 6

CREATE a program that will take as INPUT the name of a color of the spectrum (RED, ORANGE, YELLOW, GREEN, BLUE, INDIGO, or VIOLET), and print on the screen its order. For example, if you type in YELLOW, it will print 3. If you type BLUE, it will print 5.

Be sure that your program will also do the reverse. You INPUT 3, it prints YELLOW.

GREEN PROJECT 7

CREATE a graphic display of a castle with a knight moving across your screen.

GREEN PROJECT 8

CREATE graphics to go with the following verse. Use subroutines.

From the snows of winter
come the flowers of spring.
From the heat of summer
look what fall will bring.

Send us your solution to this project, and we'll send you your PROGRAMMER IV card. This GREEN page must accompany your request.

Send to:

Henry A. Taitt
CREATIVE Programming, Inc.
604 Sixth Street
Charleston, IL 61920

TI-99/4A

Your name ______________________

Phone # ________________________

Address ________________________

City, State _____________________

Zip ________ Birthdate ___________

Don't forget to enclose a self-addressed stamped envelope.

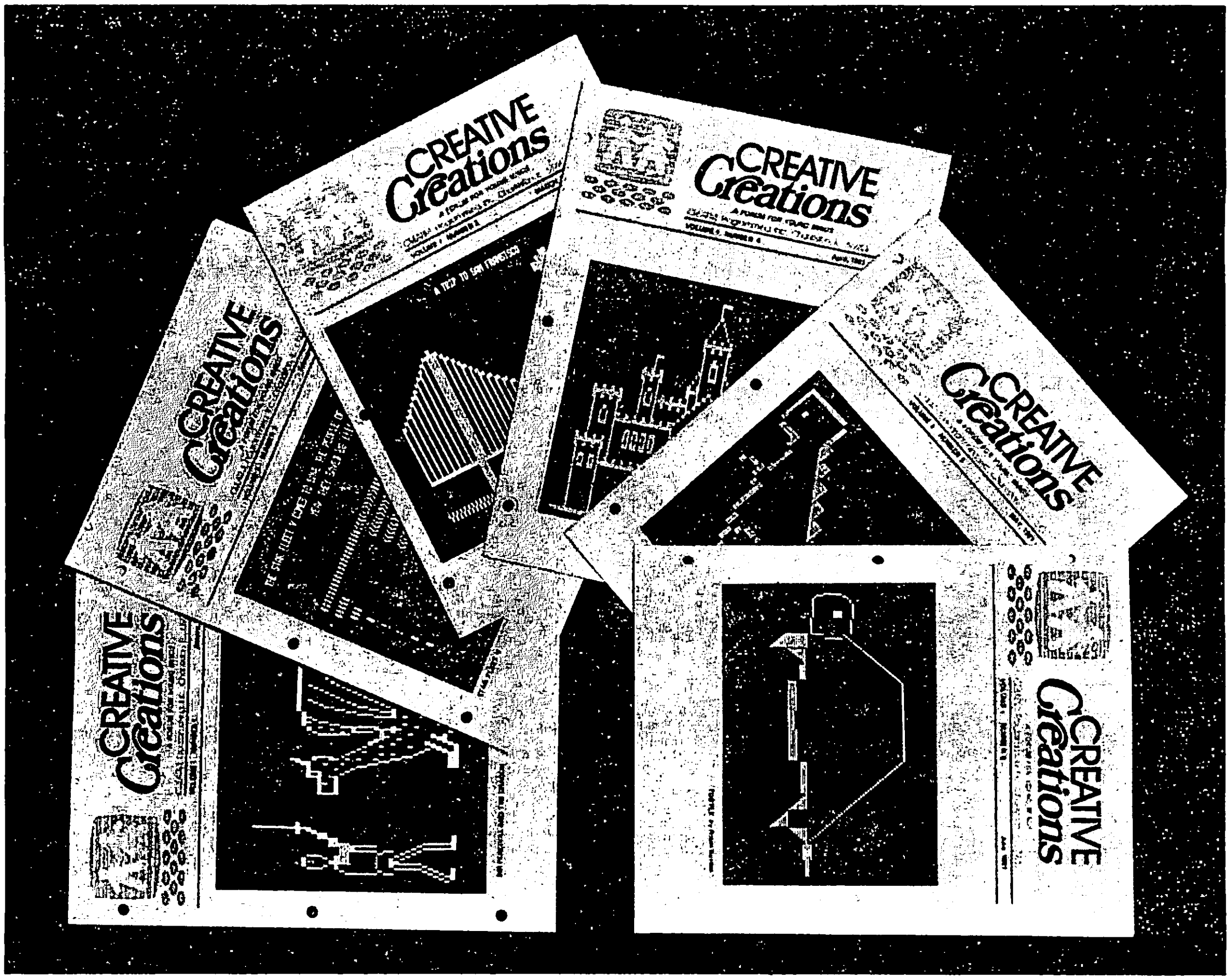
CREATIVE Creations

A FORUM FOR YOUNG MINDS

CREATIVE Programming, Inc., Charleston, IL 61920

A newsletter published 12 times a year. The articles are for young programmers, about young programmers and often written by young programmers.

Each month a graphics program created by a student is selected for the cover. It could be yours! Contests, mind bending challenges, computer game reviews, new creations, programs, even an X-rated column for parents and teachers who are running programs in their areas.



Name________________________________

Address______________________________

City____________________ State________ Zip________

Please make checks payable to: CREATIVE Creations
604 Sixth Street
Charleston, IL 61920

Only $18 a year ($32 for two years) brings all twelve issues to your door. Join us today in sharing in the excitement of CREATIVE Programming through CREATIVE Creations.

☐ one year ($18.00) ☐ two years ($32.00)

CREATIVE PROGRAMMING INCORPORATED
A SUBSIDIARY OF R.V. WEATHERFORD CO.
604 6th St., Charleston, IL 61920

TI9-R-04